MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2018/2019 SESSION

ECE3206 – OBJECT ORIENTED PROGRAMMING WITH C++

(CE, EE, ME, TE)

14 MARCH 2019
9:00 A.M – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This question paper consists of nine (9) pages only (including this page).
2. **This is an open-book exam**, so the questions are more in-depth and there are fewer questions than normal exams. Students are allowed a maximum of 20 stapled papers of reference sheets into the exam hall.
3. There are **THREE (3) QUESTIONS** in this paper. Answer **ALL QUESTIONS**. Question 1 carry 50 marks and Question 2 carries 30 marks and question 3 carries 20 marks.
4. All programming questions must be answered using the C++ language.
5. Write your answers in the Answer Booklet provided.
6. State all assumptions clearly.

Question 1

Figure Q1-1 illustrates a Unified Modeling Language (UML) class diagram for class **`Trip`**. This class reads and stores the data for 15 trips and calculates the gross profit for a particular driver.

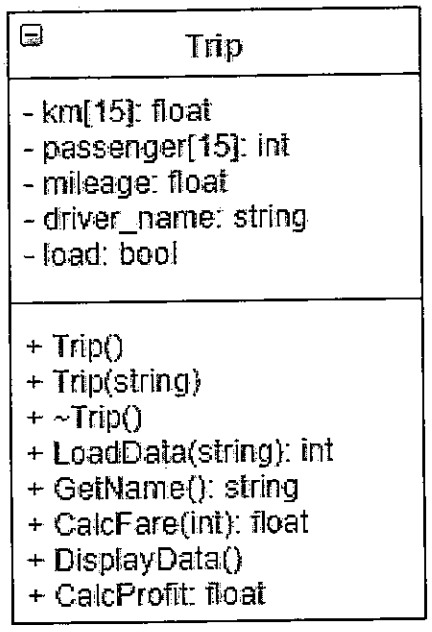

Figure Q1-1: Class `Trip` UML diagram.

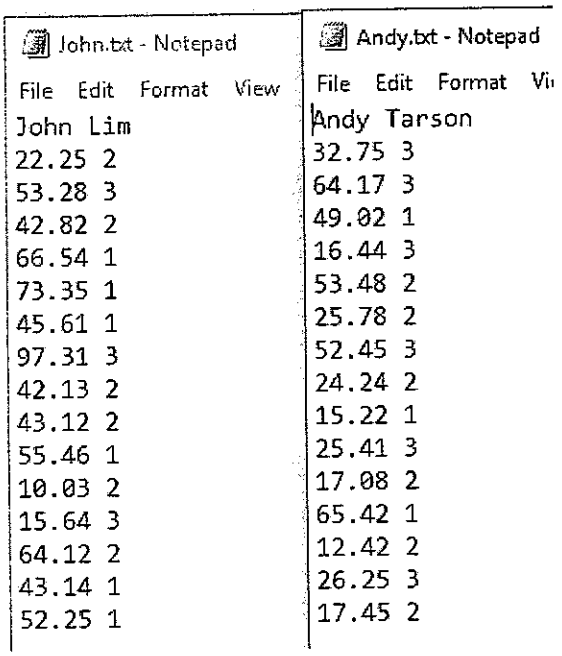

Figure Q1-2: Data text files John.txt and Andy.txt

Figure Q1-2 illustrates two data text files `John.txt` and `David.txt` containing the data of 15 trips for the two drivers John and Andy. The first line denotes the driver's name while each subsequent row contains the kilometers travelled (in km) and number of passengers.
Using C++ programming language and with reference to Figure Q1-1:

(a) Implement all the member functions, constructor and destructor of class **`Trip`**.

i. Declaration of the private members of the class according to the UML diagram.

[4 marks]

ii. The **`Trip()`** default constructor sets the private members (including all array elements) to zero, the `mileage` class variable to RM0.65 and false for Boolean variable `load`.

[4 marks]

iii. The **`Trip()`** parameterized constructor takes in the input **`fileName`** parameter to load the file. Subsequently it:

- It sets the `mileage` and `load` class variables to RM0.65 and false respectively.
- The constructor runs **`LoadData()`** method which is described below

Continued ...

- If data is successfully loaded (check the `load` class variable), then run **`DisplayData()`**
- If the data file is not successfully opened, the constructor outputs the error `"Unable to open file. Please correct check for the correct file name and reload."`.

[5 marks]

iv. The **`~Trip()`** destructor frees the objects created.

[1 marks]

v. The **`LoadData()`** member function takes in a string called `FileName` as input parameter.

- It then uses the fstream `readFile()` method to open the file.
- If the file can be opened, it will first read the driver's name in the `driver_name` class variable.
- Then it reads in up to 15 lines of data represented in two columns. The first column is the number of kilometers travelled which is read into the km array class variable. The second column representing the number of passengers is read into the `passenger` array class variable.
- Upon success reading of the file, it will close the fstream data file
- Then set the Boolean member variable `load` to TRUE, prints out the success line "*`FileName`*`: Data loaded successfully.`", where *`FileName`* is the name of the input string, and returns 1.
- If the file cannot be opened, it will print the error message for the user "*`FileName`*`: Data not loaded successfully.`" and returns 0.

[10 marks]

vi. The **`GetName()`** member function returns the value of the `driver_name` class variable.

[2 marks]

vii. The **`CalcFare()`** member function takes in an integer `i`, which is the index of the record to be calculated upon.

- Initial fare is calculated upon mileage multiplied by the kilometers travelled.
- If there is a second passenger, a RM5 surcharge is imposed.
- If there are 3 passengers, then the surcharge imposed is 15% on the initial fare. (no additional RM5 surcharge in this case.)
- The final fare is returned.

[5 marks]

viii. The **`DisplayData()`** member function takes in no input

Continued ...

- It throws the error string "No data loaded yet." if the load member variable is false.
- Otherwise it prints out the display in Figure Q1-4.

```
Driver Name:John Lim
Trip 1:22.25 km Number of Passengers:2 Earnings:RM19.4625
Trip 2:53.28 km Number of Passengers:3 Earnings:RM39.8268
Trip 3:42.82 km Number of Passengers:2 Earnings:RM32.833
Trip 4:66.54 km Number of Passengers:1 Earnings:RM43.251
Trip 5:73.35 km Number of Passengers:1 Earnings:RM47.6775
Trip 6:45.61 km Number of Passengers:1 Earnings:RM29.6465
Trip 7:97.31 km Number of Passengers:3 Earnings:RM72.7392
Trip 8:42.13 km Number of Passengers:2 Earnings:RM32.3845
Trip 9:43.12 km Number of Passengers:2 Earnings:RM33.028
Trip 10:55.46 km Number of Passengers:1 Earnings:RM36.049
Trip 11:10.03 km Number of Passengers:2 Earnings:RM11.5195
Trip 12:15.64 km Number of Passengers:3 Earnings:RM11.6909
Trip 13:64.12 km Number of Passengers:2 Earnings:RM46.678
Trip 14:43.14 km Number of Passengers:1 Earnings:RM28.041
Trip 15:52.25 km Number of Passengers:1 Earnings:RM33.9625
```

Figure Q1-3: sample output when DisplayData() is called on John.txt.

[**Hint 1**: array indexes starts from 0 whereas the trip counter starts from 1. **Hint 2**: Earnings are calculated upon calling `CalcFare()` member function above.]

[5 marks]

ix. The **`CalcProfit()`** member function takes in no input

- It throws the similar error string "No data loaded yet." if the `load` member variable is false.
- Otherwise it will calculate and return the **sum of all fares** generated by calling `CalcFare()` for all the 15 trips.

[5 marks]

(b) Write a **`main()`** function to test the implemented class **`Trip`** with proper exception handling. The sample output of the entire program is displayed in Figure 1-4.

i. Declare and initialize an object for class Trip called `driver1` using the parameterized constructor. Use the data file "John.txt" as the input parameter. Lastly display the total profits by calling `CalcProfit()` for `driver1`.

[2 marks]

ii. Declare an object for class **`Trip`** called `driver2` using the default constructor. Then, declare a string `filename` and prompt the user iteratively to key in a filename to be opened by `LoadData()` for `driver2`. The loop stops only when data is successfully loaded. Figure1-4 illustrates one failed attempt to open the file `andy.txt` and a subsequent successful attempt.

[4 marks]

Continued ...

iii. Subsequently call `DisplayData()` for `driver2` and display Andy's total profit by calling `CalcProfit()` for `driver2` within a **try-catch** block. Display any error messages caught as a string object.

[3 marks]

```
John.txt:Data loaded successfully.
Driver Name:John Lim
Trip 1:22.25 km Number of Passengers:2 Earnings:RM19.4625
Trip 2:53.28 km Number of Passengers:3 Earnings:RM39.8268
Trip 3:42.82 km Number of Passengers:2 Earnings:RM32.833
Trip 4:66.54 km Number of Passengers:1 Earnings:RM43.251
Trip 5:73.35 km Number of Passengers:1 Earnings:RM47.6775
Trip 6:45.61 km Number of Passengers:1 Earnings:RM29.6465
Trip 7:97.31 km Number of Passengers:3 Earnings:RM72.7392
Trip 8:42.13 km Number of Passengers:2 Earnings:RM32.3845
Trip 9:43.12 km Number of Passengers:2 Earnings:RM33.028
Trip 10:55.46 km Number of Passengers:1 Earnings:RM36.049
Trip 11:10.03 km Number of Passengers:2 Earnings:RM11.5195
Trip 12:15.64 km Number of Passengers:3 Earnings:RM11.6909
Trip 13:64.12 km Number of Passengers:2 Earnings:RM46.678
Trip 14:43.14 km Number of Passengers:1 Earnings:RM28.041
Trip 15:52.25 km Number of Passengers:1 Earnings:RM33.9625
John Lim's Total profit:RM:518.79
Enter file name to load:
Andy
Andy:Data not loaded successfully.
Enter file name to load:
Andy.txt
Andy.txt:Data loaded successfully.
Driver Name:Andy Tarson
Trip 1:32.75 km Number of Passengers:3 Earnings:RM24.4806
Trip 2:64.17 km Number of Passengers:3 Earnings:RM47.9671
Trip 3:49.02 km Number of Passengers:1 Earnings:RM31.863
Trip 4:16.44 km Number of Passengers:3 Earnings:RM12.2889
Trip 5:53.48 km Number of Passengers:2 Earnings:RM39.762
Trip 6:25.78 km Number of Passengers:2 Earnings:RM21.757
Trip 7:52.45 km Number of Passengers:3 Earnings:RM39.2064
Trip 8:24.24 km Number of Passengers:2 Earnings:RM20.756
Trip 9:15.22 km Number of Passengers:1 Earnings:RM9.893
Trip 10:25.41 km Number of Passengers:3 Earnings:RM18.994
Trip 11:17.08 km Number of Passengers:2 Earnings:RM16.102
Trip 12:65.42 km Number of Passengers:1 Earnings:RM42.523
Trip 13:12.42 km Number of Passengers:2 Earnings:RM13.073
Trip 14:26.25 km Number of Passengers:3 Earnings:RM19.6219
Trip 15:17.45 km Number of Passengers:2 Earnings:RM16.3425
Andy Tarson's total profit:374.63
Press any key to continue . . .
```

Figure 1-4: Complete sample output for part (b) main function. Underlined text denotes user input.

Continued ...

Question 2

(a) Create a class, **`TimeCounter`**, which keeps a time count, in terms of number of seconds. The class has one private instance variable, **`seconds`** (type **`int`**). Provide the following public members:

- A constructor that accepts an integer argument (with default value zero) for the initialization of the instance variable **`seconds`**.
- Functions **`setSecondsCount`** and **`getSecondsCount`** to set and get the value of the instance variable.
- A function, **`getTimeCount`**, which returns a string representation of the time count, in minutes and seconds. For example, for a time count of 125 seconds, this function should return a string, "2m 5s".

[8 marks]

(b) Create an abstract class, **`Playable`**, which represents any entity that may be played. The class has two instance variables of **`protected`** access, **`title`** (type **`string`**) and **`duration`** (type **`TimeCounter`**). Provide the following public members:

- A constructor accepting appropriate arguments for the initialization of the class's instance variables.
- Set functions for each of the instance variables. The variable **`duration`** requires a new instance of **`TimeCounter`** to be created.
- A pure virtual function, **`getSummary`**, which doesn't require any arguments and returns a **`string`** type.

[7 marks]

Continued ..

(c) Derive two classes from class **`Playable`**: **`Song`** and **`Audiobook`**. For each of the derived classes, provide a constructor and an implementation for the base class's pure virtual function (note: for simplicity's sake, you are not required to write the setter and getter functions).

- Class **`Song`**: defines 2 instance variables, **`artist`** (type **`string`**) and **`year`** (type **`int`**). The constructor accepts appropriate arguments for the initialization of all instance variables, including the inherited ones. The function **`getSummary`** returns a string in the format:

	Example:
***title* (*duration*)**	`Close to You (3m 43s)`
By: *artist, year*	`By: Carpenters, 1970`

- Class **`Audiobook`**: defines 2 instance variables of type **`string`**, **`author`** and **`narrator`**. The constructor accepts appropriate arguments for the initialization of all instance variables, including the inherited ones. The function **`getSummary`** returns a **`string`** in the format:

	Example:
***title* (Audiobook)**	`Jane Eyre (Audiobook)`
By: *author*	`By: Charlotte Bronte`
Narrated by: *narrator*	`Narrated by: Elizabeth Klett`
Length: *duration*	`Length: 1106m 1s`

[12 marks]

(d) Given the code snippet in Figure Q2.1, describe the possible output from the **`cout`** line, and explain the observable polymorphic behaviour.

```
Playable *list[3];

/*codes for initialization of array elements*/

for (int i=0; i<3; i++)
   cout << list[i]->getSummary() << endl;
```

Figure Q2.1

[3 marks]

Continued ..

Question 3

(a) Write a template function **`average()`** that returns the average of all the elements of an array. The arguments to the function should be the pointer to the array and the size of the array. In **`main()`**, test the function with arrays of type int and float. Each array has 4 values and the result should be displayed.

[10 Marks]

(b) Define a class template `myArray` to represent a generic array that can store different type of data. The class should contain member function `setArray(int elem, T val)` that store the value `val` at position `elem` in the array. Member function **`getArray()`** display all elements in the array. The constructor should allocate memory for the array. Use the driver function in the Figure Q3.1 and sample program output at Figure Q3.2 as your reference. You are required to declare the class `MyArray` and implement all the required member functions.

[10 Marks]

```
int main()
{
// instantiate int_array object of class array<int> with size 2
MyArray< int > int_array(2);
// set value to a first element
int_array.setArray(0,3);
// set value to a second element
int_array.setArray(1,4);
// call to array class member function to display array elements
int_array.getArray();
cout<< "\n";

// instantiate float_array object of class array<float> with size 2
MyArray< float > float_array(2);
// set value to a first element
float_array.setArray(0,3.9);
// set value to a second element
float_array.setArray(1,2.9);
float_array.getArray();

return 0;
}
```

Figure Q3.1

Continued ...

```
Value at position 0: 3
Value at position 1: 4

Value at position 0:  3.9
Value at position 1:  2.9
```

Figure Q3.2

End of Paper